Available online at www.sciencedirect.com

SciVerse ScienceDirect

Procedia Engineering 15 (2011) 3406 – 3410

Procedia Engineering

www.elsevier.com/locate/procedia

Advanced in Control Engineering and Information Science

Highly Resilient Minimal Path Routing Algorithm for Fault Tolerant Network-on-Chips

Ka Lok Man[a], Karthik Yedluri[b], Hemangee K. Kapoor[b], Chi-Un Lei[c], Eng Gee Lim[a], Jieming Ma[a,d,*]

[a]*Xi'an Jiaotong-Liverpool University, No.111 Ren'ai Road , Suzhou 215 123, CHN,*
[b]*Indian Institute of Technology Guwahati, Assam, India.*
[c]*University of Hong Kong, Pokfulam, Hong Kong, CHN*
[d]*Univeristy of Liverpool, Department of Computer Science, Liverpool L69 3BX, UK*

Abstract

We develop a routing algorithm for fault tolerant 2-D mesh Network-on-Chips (NoCs) with permanent faults. The proposed approach is adaptive and distributed, and does not require extra circuitry or routing tables for fault tolerance operation. Deadlock handling and multiple hop links checking are included for a robust system operation. We demonstrate the algorithm mechanism using a mesh example.

Keyworks: Network-on-Chips; Fault Tolerant; Routing;

1. Introduction

Network-on-Chip (NoC) is an approach to design the communication subsystem between IP cores in a System-on-Chip (SoC) [1]. NoC applies networking theory and algorithms to on-chip communications. As the chip manufacturing technology scales down to the nanometer domain, the probability of error increases [2]. Smaller feature sizes and higher frequencies introduce transient faults, which are data errors. Also, manufacturing defects and device wear out effects lead to permanent faults (broken physical links and/or routers) and cause an entire NoC to fail. Hence, fault tolerance of the network on chip has become

* Corresponding author: Jieming Ma. Tel.: +86-512-88161831.
E-mail address:Jieming@liverpool.ac.uk; Jieming.Ma08@student.xjtlu.edu.cn.

1877-7058 © 2011 Published by Elsevier Ltd. Open access under CC BY-NC-ND license.
doi:10.1016/j.proeng.2011.08.638

very important. In this context, fault tolerance is the ability to route packets in presence of faults in the network with certain degree of tolerance.

When there are faults in routing paths, hardwares cannot forward the packets further. However, it is not possible to replace the faulty components on the chip. Hence, the routing algorithm must be updated to adapt to the changes at runtime. NoC inherently has redundant paths between IP cores, and is able to build a reliable system. However, fault tolerance routing guarantees the packet delivery, but cannot ensure Quality of Service (QoS). Any routing algorithm can be fully fault tolerant only at the expense of QoS.

In this paper, we propose an adaptive, distributed and fault tolerant routing algorithm for 2-D mesh NoCs with permanent faults (e.g. physical link and node faults). Comparing with existing algorithms, our algorithm does not need any extra circuitry or routing tables for fault tolerance operation. Furthermore, our work has no restrictions on the fault patterns, therefore no active node is deactivated for fault pattern requirements. After the literature review in Section 2, the mechanism of the proposed algorithm is presented in Section 3. Numerical examples in Section 4 then confirm the performance of the algorithm.

2. Related Work

Many fault tolerant routing approaches have already been proposed for on-chip networks [3], [4], [6], [7], [8]. The approach in [3] uses spare wires (hardware redundancy) to bypass the faulty links. However, use of extra circuitry for fault tolerance leads to area overhead. Furthermore, fault tolerance methods in [3] use different methods for error detection and error recovery, with expense of extra circuitry. Some approaches (e.g. [4]) use a routing table at each router. Routers update their routing tables when faults appear in the network. Hence, router complexity is increased. This requires more memory in the routers to store the routing tables. Updating routing tables is time consuming, sometimes gives inconsistent results and introduces more delay. The works presented in [5] adaptively route packets around faulty routers. However, they put various restrictions on the fault patterns. Many traditional algorithms deactivate a number of active nodes to satisfy the fault patterns. This will result in a traffic congestion.

3. Proposed Fault Tolerant Routing

We present an adaptive, distributed and fault tolerant routing algorithm for 2-D mesh NoCs, which reroutes packets around permanent faulty components. With each destination node as center, the network is divided into quadrants called destination quadrants. Number of destination quadrants for a destination can be 1, 2 or 4, depending on the position of the destination node in the mesh. The pseudocodes of the proposed fault tolerance routing and the Route function are shown in Algorithm 1 and 2, respectively.

3.1. Routing Judgment

At any arbitrary router, the next-hop decision is a function of current node position in destination quadrants and the input direction of the packet. Allowed output directions are different for the same input direction in all destination quadrants. Table 1 shows routing directions at a node in all destination quadrants as a function of input direction. The output direction depends on the position of the current node. Each router runs an instance of routing logic independently. For a given input direction, routers take the first allowed output direction and check for 1-hop link faults. When 1-hop link is not faulty, all 2-hop links in the same direction are checked. In contrast, if all the paths through first output direction are faulty (i.e, either 1-hop link is faulty or all 2-hop links are faulty), the second output direction allowed is chosen. For the second output direction, 1-hop and 2-hop links are checked in the same manner as done for the first output direction. If second output direction has no fault-free path, then the third allowed output

direction is chosen with the same fault checking as above. If all the outgoing paths at the router are faulty, the packet will be routed back in the input direction it came from. This back routing is done hop by hop basis until it finds an alternative path. Back routed packet will take a longer path than the normal path.

```
Input: ip_dir, id, dest_id
Output: output_direction
Begin
  if C_X<D_X & C_Y<D_Y then
  | Refer to column 1 in the Table I for output directions.
  else
  |  if C_X<D_X & C_Y>D_Y then
  |  | Refer to column 2 in the Table I for output
  |  | directions.
  |  else
  |  |  if C_X>D_X & C_Y<D_Y then
  |  |  | Refer to column 3 in the Table I for output
  |  |  | directions.
  |  |  else
  |  |  |  if C_X>D_X & C_Y>D_Y then
  |  |  |  | Refer to column 4 in the Table I for
  |  |  |  | output directions.
  |  |  |  else
  |  |  |  |  if C_X==D_X & C_Y<D_Y then
  |  |  |  |  | Refer to column 5 in the Table I for
  |  |  |  |  | output directions.
  |  |  |  |  else
  |  |  |  |  |  if C_X==D_X & C_Y>D_Y then
  |  |  |  |  |  | Refer to column 6 in the Table I
  |  |  |  |  |  | for output directions.
  |  |  |  |  |  else
  |  |  |  |  |  |  if C_Y==D_Y & C_X<D_X then
  |  |  |  |  |  |  | Refer to column 7 in the
  |  |  |  |  |  |  | Table I for output directions.
  |  |  |  |  |  |  else
  |  |  |  |  |  |  |  Refer to column 8 in the
  |  |  |  |  |  |  |  Table I for output directions.
  |  |  |  |  |  |  end
  |  |  |  |  |  end
  |  |  |  |  end
  |  |  |  end
  |  |  end
  |  end
  end
  return output_direction
End:
```

Algorithm 1: Pseudocodes of the proposed *Route* function

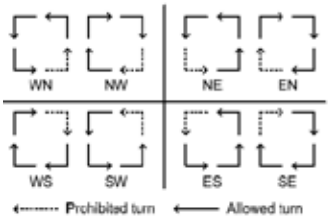


Fig. 1. Turn model: Turns taken in different destination quadrants for both clockwise and anti-clockwise directions

```
Input: ip_dir, id, dest_id
Output: next_hop
Begin
  (C_X,C_Y): Coordinates of Current node;
  (D_X,D_Y): Coordinates of Destination node;
  C_X=id%num_cols;
  C_Y=id/num_cols;
  D_X=dest_id%num_cols;
  D_Y=dest_id/num_cols;
  if C_X==D_X & C_Y==D_Y then
  |  return core
  else
  |  next_hop=Route(ip_dir, id, dest_id)
  end
  return next_hop
End:
```

Algorithm 2: Pseudocodes of the proposed fault tolerance routing

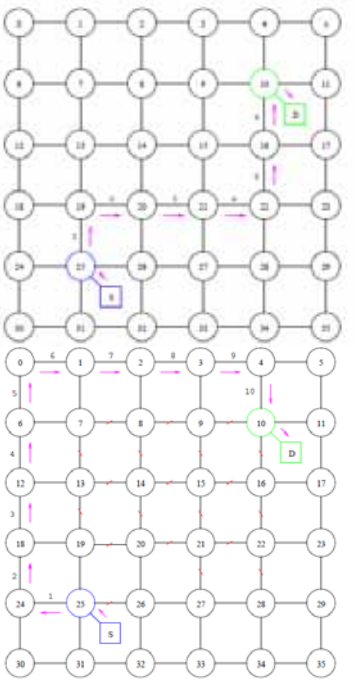


Fig. 2. Configuration for the situation: (a) with zero fault; (b) with faulty paths

From any node in 2-D mesh, packets generally have two output directions leading towards destination and one output direction going away from destination. Paths in those two output directions leading towards destination are checked first. The packet traversal path to destination through these directions is minimum, when all links in the path are fault free. Otherwise, if the router finds any link faulty, the packet will be routed in an alternative direction. In this case, the packet may take a slightly longer path. In

the worst case, the packet takes the third output direction which obviously leads to the longest path. However, the delivery of the packet is always guaranteed if there is an existing path to the destination.

Table 1. Output directions in different destination quadrants (N: North, E: East, S: South, W: West)

Input direction	$C_X<D_X$ $C_Y<D_Y$	$C_X<D_X$ $C_Y>D_Y$	$C_X>D_X$ $C_Y<D_Y$	$C_X>D_X$ $C_Y>D_Y$	$C_X==D_X$ $C_Y<D_Y$	$C_X==D_X$ $C_Y>D_Y$	$C_Y==D_Y$ $C_X<D_X$	$C_Y==D_Y$ $C_X>D_X$
North	SE	ESW	SW	WSE	SE	SWE	ES	WSE
West	ES	EN	SEN	NES	SE	NES	ES	ENS
South	ENW	NE	WNE	NW	NEW	NW	ENW	WN
East	SWN	NWS	WS	WN	SWN	NW	WSN	WN
Core	ESNW	NEWS	WSEN	WNSE	SEWN	NWES	ESNW	WNSE

Re-routing packets around faulty components may form cycle in the network leading to deadlocks. Some packets in the network are then blocked and will stay blocked forever due to these deadlocks. Virtual channels [5] can be used to avoid deadlocks, but the implementation of virtual channels requires additional logic gates and hardware circuitry. This algorithm uses the turn model proposed by Glass and Ni [11]. This turn model eliminates deadlocks by prohibiting two turns in each destination quadrant. These two prohibited turns (one for clockwise direction and the other for anti-clockwise direction) avoid cycles formed. Prohibited turns in different destination quadrants are shown below in Fig. 1.

3.2. Multiple Hop Links Checking

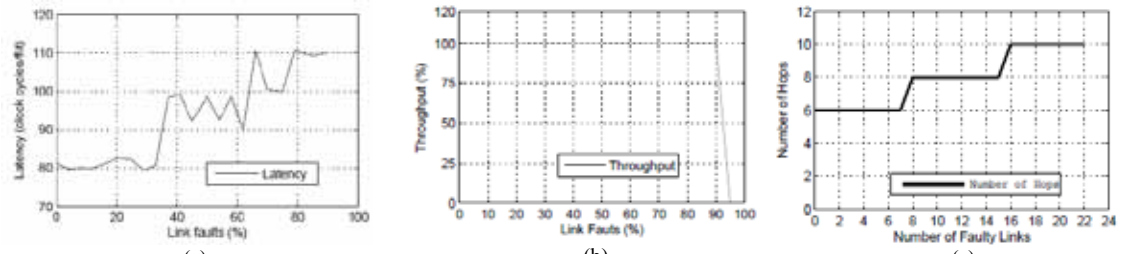
The uniqueness of our fault tolerant routing algorithm lies in the future faults prediction in allowed output directions at the current node. At present, only 1-hop and 2-hop links are checked. This can be extended to n-hop links check. These future predictions introduce delay in routing decisions made. However, packet never takes the path to fully faulty nodes, in which case the packet is routed back to the previous node. This reduces the number of hops that the packet travels.

4. Experimental Evaluation

NIRGAM, a NoC discrete event, cycle accurate simulator [10], is used for the experimental analysis. The maximum number of link faults is defined as the total number of links excluding the number of links in the path to destination. Our algorithm is deadlock free because no set of turns used in the network routing form a loop, as shown in Fig. 1.

In this example, we used a 6×6 2D mesh for simulation. The numbers of source and destination nodes are 25 and 10, respectively. The boundary between source and destination nodes forms a 4×4 mesh. Hence, the total numbers of links in the boundary and the whole mesh are 24 and 60, respectively. The algorithm checks for paths in the boundary first. If no path exists within the boundary, packets will be routed through out of boundary paths. It is because paths taken in the boundary are always minimal, and out of boundary paths are usually longer paths. However, the next minimal path direction also results in a less packet latency. In the situation with zero fault, minimal path in the boundary takes 6 hops to the destination, as shown in Fig. 2. Also, we have checked for 20 out of 24 faults in the boundary. The result is shown in Fig. 3. In this situation, all paths in the boundary are faulty. So, packets require 10 hops to reach the destination for out of boundary routing, which is the next minimal path. Next, we explored the relationship between the link fault rate and latency of flits in the network. We injected faults randomly

into the network. As shown in Fig. 3(a), the initial latency under zero fault is 81.5 clock cycles. Latency increases with the link fault rate because a longer path is taken. We also studied the relationship between the link fault rate and the throughput of the network. Our results, as shown in Fig. 3(b), show a 100% throughput when at least one path exists to destination. Throughput becomes zero when the link fault rate is 95%, i.e., faults partition the network, and packets have no path to reach the destination node. This violates our basic assumptions. At last, we investigated the effect of link failures on the number of hops taken by the packets to reach destination, and the result is shown in Fig. 3(c). Packets traverse more number of hops as the faults increases. Although this increases the latency, throughput remains the same.



(a) (b) (c)

Fig. 3. Performance analysis: (a) packet latency for different link fault ratios; (b) relationship between throughput and link faults; (c) relationship between the number of hops that is traversed by the packets to reach the destination and the number of faulty links

5. Conclusion

A fault tolerant routing algorithm has been presented for 2-D mesh NoCs. It has been shown that the algorithm is adaptive and distributed. Deadlock handling and multiple hop links checking have been included for a robust system operation. A routing example has confirmed the algorithm performance.

References

[1] Benini L, Micheli GD. Networks on Chips: a New SoC Paradigm. *IEEE Computer* 2002; **35**: 70-78.

[2] Dumitras T, Kerner S, Marculescu R. Towards on-chip fault-tolerant communication. *Proc. IEEE ASP-DAC*, January 2003, p. 225-232.

[3] Lehtonen T, Liljeberg Pand Plosila J. Online reconfigurable self-timed links for fault tolerant NoC. VLSI Design, November 2007, Article ID 94676, 13 pages.

[4] Fick D, DeOrio A, Chen G, Bertacco V, Sylvester D, Blaauw D. A highly resilient routing algorithm for fault-tolerant NoCs. *Proc. IEEE DATE*, April 2009, p. 21-26.

[5] Rameshan N, Ahmed M, Gaur MS, Laxmi V. Minimal path, Fault Tolerant, QoS aware Routing with node and link failure in 2-D Mesh NoC. *Proc. IEEE DFT*, October 2010, p. 60-66.

[6] Patooghy A, Miremadi SG. XYX: A power & performance efficient fault-tolerant routing algorithm for network on chip. *Proc. IEEE ICPDNP*, February 2009, p. 245-251.

[7] KimYB. Fault tolerant source routing for network-on-chip. *Proc. IEEE DFT*, October 2007, p. 12-20.

[8] Zhang Z, Greiner A, Taktak S. A reconfigurable routing algorithm for a fault-tolerant 2D-mesh Network-on-Chip. *Proc. ACM/IEEE DAC*, June 2008, p. 441-446.

[9] Feng C, Lu Z, Jantsch A, Li J, Zhang M. A reconfigurable fault-tolerant deflection routing algorithm based on reinforcement learning for network-on-chip. *Proc. ACM NoCArc*, 2010, p. 11-16.

[10] Jain L, Al-Hashimi BM, M.S. Gaur, Laxmi V. NIRGAM: A Simulator for NoC Interconnect Routing and Aplication Modelling. *Proc. IEEE DATE*, April 2007.

[11] Glass CJ, Ni LM. The turn model for adaptive routing. *Journal of the ACM* 1994; **41**:874-902.